Terms used:
hot dynamic quiescient swapping translation near5 operating system

Found **10,385** of **205,978**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown

Relevance scale

1 Type-based hot swapping of running modules (extended abstract)

Dominic Duggan
October 2001 **ACM SIGPLAN Notices , Proceedings of the sixth ACM SIGPLAN international conference on Functional programming ICFP '01**, Volume 36 Issue 10
Publisher: ACM Press

Full text available: pdf(150.34 KB) Additional Information: full citation, abstract, references, citings, index terms

While dynamic linking has become an integral part of the run-time execution of modem programming languages, there is increasing recognition of the need for support for hot swapping of running modules, particularly in long-lived server applications. The interesting challenge for such a facility is to allow the new module to change the types exported by the original module, while preserving type safety. This paper describes a type-based approach to hot swapping running modules. The approach is bas ...

Keywords: dynamic typing, hot swapping, module interconnection languages, shared libraries

2 Optimising hot paths in a dynamic binary translator

David Ung, Cristina Cifuentes
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Publisher: ACM Press

Full text available: pdf(890.10 KB) Additional Information: full citation, abstract, citings, index terms

In dynamic binary translation, code is translated "on the fly" at run-time, while the user perceives ordinary execution of the program on the target machine. Code fragments that are frequently executed follow the same sequence of flow control over a period of time. These fragments form a hot path and are optimised to improve the overall performance of the program.Multiple hot paths may also exist in programs. A program may choose to execute in one hot path for some time, but later switch to anot ...

Keywords: binary translation, dynamic compilation, dynamic execution, run-time profiling

3 Dynamic applications from the ground up

Don Stewart, Manuel M. T. Chakravarty
September 2005 **Proceedings of the 2005 ACM SIGPLAN workshop on Haskell Haskell '05**
Publisher: ACM Press

Full text available: pdf(188.51 KB) Additional Information: full citation, abstract, references, citings, index terms

Some Lisp programs such as Emacs, but also the Linux kernel (when fully modularised) are *mostly dynamic;* i.e., apart from a small static core, the significant functionality is dynamically loaded. In this paper, we explore *fully dynamic* applications in Haskell where the static core is minimal and code is hot swappable. We demonstrate the feasibility of this architecture by two applications: Yi, an extensible editor, and Lambdabot, a plugin-based IRC robot. Benefits of the approach i ...

Keywords: dynamic applications, dynamic update, extension languages, functional programming, hot swapping

4 Machine-adaptable dynamic binary translation

David Ung, Cristina Cifuentes
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization DYNAMO '00**, Volume 35 Issue 7
Publisher: ACM Press

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

5 Dynamic and adaptive updates of non-quiescent subsystems in commodity operating system kernels

Kristis Makris, Kyung Dong Ryu
March 2007 **ACM SIGOPS Operating Systems Review , Proceedings of the 2007 conference on EuroSys EuroSys '07**, Volume 41 Issue 3
Publisher: ACM Press

Full text available: pdf(452.03 KB) Additional Information: full citation, abstract, references, index terms

Continuously running systems require kernel software updates applied to them without downtime. Facilitating fast reboots, or delaying an update may not be a suitable solution in many environments, especially in pay-per-use high-performance computing clusters and mission critical systems. Such systems will not reap the benefits of new kernel features, and will continue to operate with kernel security holes unpatched, at least until the next scheduled maintenance downtime. To address these prob ...

Keywords: DynAMOS, adaptive operating system, dynamic instrumentation, dynamic software updates, function cloning

6 Type-Safe linking with recursive DLLs and shared libraries

Dominic Duggan
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Publisher: ACM Press

Full text available: pdf(658.62 KB) Additional Information: full citation, abstract, references, citings, index terms

Component-based programming is an increasingly prevalent theme in software development, motivating the need for expressive and safe module interconnection languages. Dynamic linking is an important requirement for module interconnection languages, as exemplified by dynamic link libraries (DLLs) and class loaders in operating systems and Java, respectively. A semantics is given for a type-safe module interconnection language that supports shared libraries and dynamic linking, as well as circular ...

Keywords: Dynamic Linking, Module Interconnection Languages, Recursive Modules, Shared Libraries

7 UTLB: a mechanism for address translation on network interfaces

Yuqun Chen, Angelos Bilas, Stefanos N. Damianakis, Cezary Dubnicki, Kai Li
October 1998 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the eighth international conference on Architectural support for programming languages and operating systems ASPLOS-VIII**, Volume 33 , 32 Issue 11 , 5
Publisher: ACM Press

Full text available: pdf(1.76 MB) Additional Information: full citation, abstract, references, citings, index terms

An important aspect of a high-speed network system is the ability to transfer data directly between the network interface and application buffers. Such a *direct data path* requires the network interface to "know" the virtual-to-physical address translation of a user buffer, *i.e.*, the physical memory location of the buffer. This paper presents an efficient address translation architecture, User-managed TLB (UTLB), which eliminates system calls and device interrupts from the common co ...

8 Dynamic software updating

Michael Hicks, Scott Nettles
November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6
Publisher: ACM Press

Full text available: pdf(622.69 KB) Additional Information: full citation, abstract, references, index terms

Many important applications must run continuously and without interruption, and yet also must be changed to fix bugs or upgrade functionality. No prior general-purpose methodology for dynamic updating achieves a practical balance between flexibility, robustness, low overhead, ease of use, and low cost.We present an approach for C-like languages that provides type-safe dynamic updating of native code in an extremely flexible manner---code, data, and types may be updated, at programmer-determined ...

Keywords: Dynamic software updating, typed assembly language

9 GPGPU: general purpose computation on graphics hardware

David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn
August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**
Publisher: ACM Press
Full text available: pdf(63.03 MB) Additional Information: full citation, abstract, citings

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

10 Continuous program optimization: A case study

Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4
Publisher: ACM Press
Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

Keywords: Dynamic code generation, continuous program optimization, dynamic reoptimization

11 Resource partitioning in general purpose operating systems: experimental results in Windows NT

D. G. Waddington, D. Hutchison
October 1999 **ACM SIGOPS Operating Systems Review**, Volume 33 Issue 4
Publisher: ACM Press
Full text available: pdf(1.56 MB) Additional Information: full citation, abstract, citings, index terms

The principal role of the operating system is that of resource management. Its task is to present a set of appropriate services to the applications and users it supports. Traditionally, general-purpose operating systems, including Windows NT, federate resource sharing in a fair manner, with the predominant goal of efficient resource utilisation. As a result the chosen scheduling algorithms are not suited to applications that have stringent Quality-of-Service (QoS) and resource management require ...

12 Profile-based optimizations: Dynamic trace selection using performance monitoring hardware sampling

Howard Chen, Wei-Chung Hsu, Jiwei Lu, Pen-Chung Yew, Dong-Yuan Chen
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**
Publisher: IEEE Computer Society
Full text available: pdf(1.88 MB) Additional Information: full citation, abstract, references, citings, index terms

Optimizing programs at run-time provides opportunities to apply aggressive optimizations to programs based on information that was not available at compile time. At run time,

programs can be adapted to better exploit architectural features, optimize the use of dynamic libraries, and simplify code based on run-time constants.Our profiling system provides a framework for collecting information required for performing run-time optimization. We sample the performance hardware registers available on ...

13 Efficient management for large-scale flash-memory storage systems with resource conservation

Li-Pin Chang, Tei-Wei Kuo
November 2005 **ACM Transactions on Storage (TOS)**, Volume 1 Issue 4
Publisher: ACM Press
Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, index terms

Many existing approaches on flash-memory management are based on RAM-resident tables in which one single granularity size is used for both address translation and space management. As high-capacity flash memory is becoming more affordable than ever, the dilemma of how to manage the RAM space or how to improve the access performance is emerging for many vendors. In this article, we propose a tree-based management scheme which adopts multiple granularities in flash-memory management. Our objective ...

Keywords: Flash memory, consumer electronics, embedded systems, memory management, portable devices, storage systems

14 Avoiding conflict misses dynamically in large direct-mapped caches

Brian N. Bershad, Dennis Lee, Theodore H. Romer, J. Bradley Chen
November 1994 **ACM SIGPLAN Notices , ACM SIGOPS Operating Systems Review , Proceedings of the sixth international conference on Architectural support for programming languages and operating systems ASPLOS-VI**, Volume 29 , 28 Issue 11 , 5
Publisher: ACM Press
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes a method for improving the performance of a large direct-mapped cache by reducing the number of conflict misses. Our solution consists of two components: an inexpensive hardware device called a Cache Miss Lookaside (CML) buffer that detects conflicts by recording and summarizing a history of cache misses, and a software policy within the operating system's virtual memory system that removes conflicts by dynamically remapping pages whenever large numbers of conflict miss ...

15 ASHs: application-specific handlers for high-performance messaging

Deborah A. Wallach, Dawson R. Engler, M. Frans Kaashoek
August 1997 **IEEE/ACM Transactions on Networking (TON)**, Volume 5 Issue 4
Publisher: IEEE Press
Full text available: pdf(174.62 KB) Additional Information: full citation, references, index terms

Keywords: computer networks, dynamic code generation, modular computer systems, operating systems, protocols, software protection, user-level networking

16 ASHs: Application-specific handlers for high-performance messaging

Deborah A. Wallach, Dawson R. Engler, M. Frans Kaashoek
August 1996 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Applications, technologies, architectures, and protocols**

for computer communications SIGCOMM '96, Volume 26 Issue 4
Publisher: ACM Press

Full text available: pdf(174.50 KB) Additional Information: full citation, abstract, references, citings, index terms

Application-specific safe message handlers (ASHs) are designed to provide applications with hardware-level network performance. ASHs are user-written code fragments that safely and efficiently execute in the kernel in response to message arrival. ASHs can direct message transfers (thereby eliminating copies) and send messages (thereby reducing send-response latency). In addition, the ASH system provides support for dynamic integrated layer processing (thereby eliminating duplicate message ...

17 Embedded systems: Arithmetic-based address translation for energy-efficient virtual memory support in low-power, real-time embedded systems

Xiangrong Zhou, Peter Petrov
September 2005 **Proceedings of the 18th annual symposium on Integrated circuits and system design SBCCI '05**
Publisher: ACM Press
Full text available: pdf(267.86 KB) Additional Information: full citation, abstract, references, index terms

In this paper, we present an arithmetic-based address translation scheme for low-power and real-time embedded processors with virtual memory support. General-purpose virtual memory support comes with its fundamental disadvantages of excessive power consumption and nondeterministic execution times. These disadvantages have been the main reason for not adopting virtual memory and its associated benefits in embedded systems where energy efficiency and real-time operations are major requirements. To ...

18 Devirtualizable virtual machines enabling general, single-node, online maintenance

David E. Lowell, Yasushi Saito, Eileen J. Samberg
October 2004 **ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 32 , 38 , 39 Issue 5 , 5 , 11
Publisher: ACM Press

Full text available: pdf(174.01 KB) Additional Information: full citation, abstract, references, citings, index terms

Maintenance is the dominant source of downtime at high availability sites. Unfortunately, the dominant mechanism for reducing this downtime, cluster rolling upgrade, has two shortcomings that have prevented its broad acceptance. First, cluster-style maintenance over many nodes is typically performed a few nodes at a time, mak-ing maintenance slow and often impractical. Second, cluster-style maintenance does not work on single-node systems, despite the fact that their unavailability during mainte ...

Keywords: availability, online maintenance, planned downtime, virtual machines

19 Charles W. Bachman interview: September 25-26, 2004; Tucson, Arizona

Thomas Haigh
January 2006 **ACM Oral History interviews**
Publisher: ACM Press
Full text available: pdf(761.66 KB) Additional Information: full citation, abstract

Charles W. Bachman reviews his career. Born during 1924 in Kansas, Bachman attended high school in East Lansing, Michigan before joining the Army Anti Aircraft Artillery Corp, with which he spent two years in the Southwest Pacific Theater, during World War II. After his discharge from the military, Bachman earned a B.Sc. in Mechanical Engineering in 1948, followed immediately by an M.Sc. in the same discipline, from the University of

Pennsylvania. On graduation, he went to work for Do ...

20 Memory hierarchy: Compiler-decided dynamic memory allocation for scratch-pad based embedded systems

Sumesh Udayakumaran, Rajeev Barua

October 2003 **Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems CASES '03**

Publisher: ACM Press

Full text available: pdf(213.48 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a highly predictable, low overhead and yet dynamic, memory allocation strategy for embedded systems with scratch-pad memory. A *scratch-pad* is a fast compiler-managed SRAM memory that replaces the hardware-managed cache. It is motivated by its better real-time guarantees vs cache and by its significantly lower overheads in energy consumption, area and overall runtime, even with a simple allocation scheme [4].Existing scratch-pad allocation methods are of two types. Firs ...

Keywords: compiler, embedded systems, memory allocation, scratch-pad